

Fig. 2

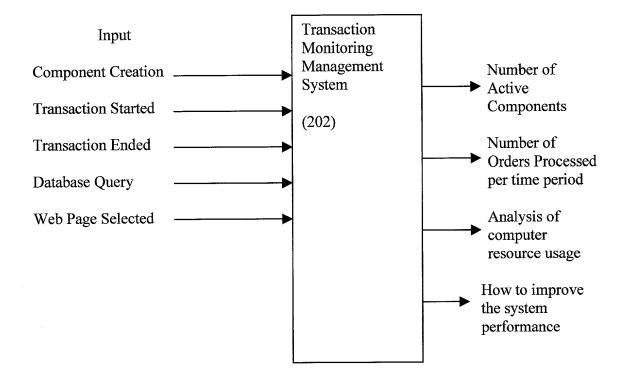


Fig. 2B

Activity

Events Transactions

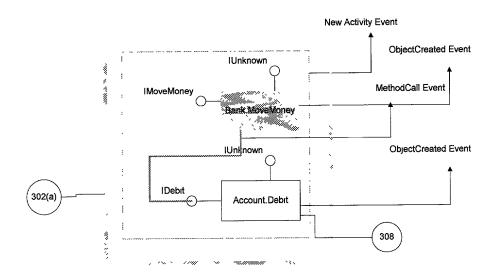
Business Model

Transfer Funds Transaction Withdrawl Transaction 1 lUnknown lUnknown IMoveMoney 306 **IWithdrawl** 320 Bank.Withdrawl 302 IUnk nown 308 **IDebit** Account Debit IDebit 322 Account.Debit 304 lUnknown lUnknown **ICredit** 310 Account.Credit **IReceipt** lUnknown 324 Bank.Receipt **IReceipt** 312 Bank.Receipt

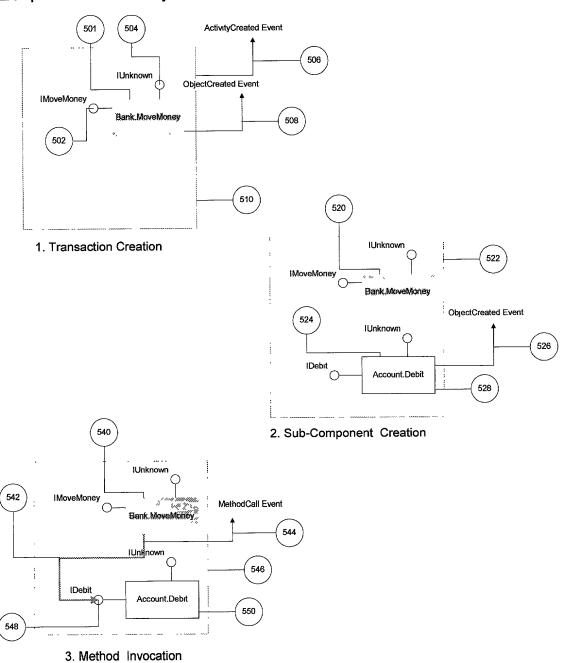
Figure 3: Transaction Object Environment

406 402 408 New Activity Event ObjectCreated Event ActivityID: 1 AcitivityID: 1 404 410 ProgID Bank.MoveMoney ObjectID: 1 412 ObjectCreated Event Method Call Event 414 422 ObjectID 2 AcıtivityID⁻ 1 InterfaceID: IDebit ProgID: Account Debit 416 424 Method: DebitAccount ObjectID: 2 418 426 420 428

Figure 4: Correlation of Transaction Object Events



Examples of Transaction Object Events - FIG 5



Application Process

Probe

Transaction Object Environment

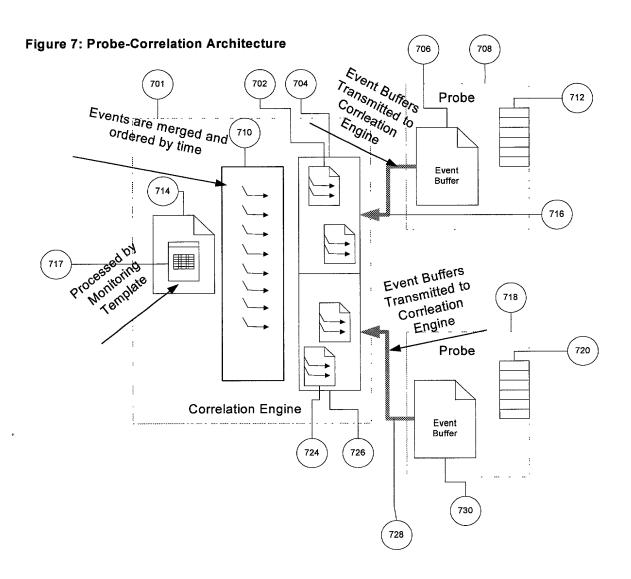
Application Process

Application Process

Frobe

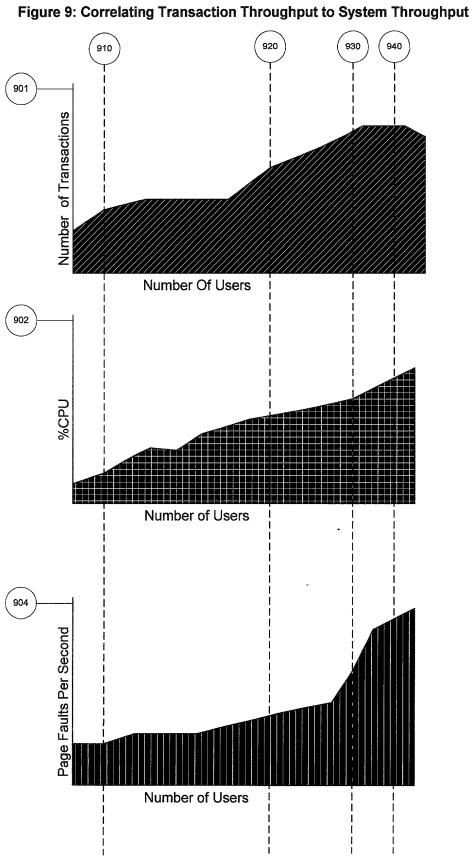
Transaction Object Environment

Figure 6: Collection of Transaction Object Events



Application Probe Probe 806 804 8 Transaction Object 802 Subscribers Environment 809 801 810 808 Application Activity Activity 816 Events Get Buffered for Fransmission Component Thread 812 814 Method Events Flow Over Transaction 818 Component Identity 820 Secunty 822 824 Method Resource ObjectCreated Event ActivityCreated Event 826 State of the state User 828 ObjectCreated Event 830 1 MoveMoney Bank.MoveMoney 832 **Event Buffer**

Figure 8: Probe Architecture



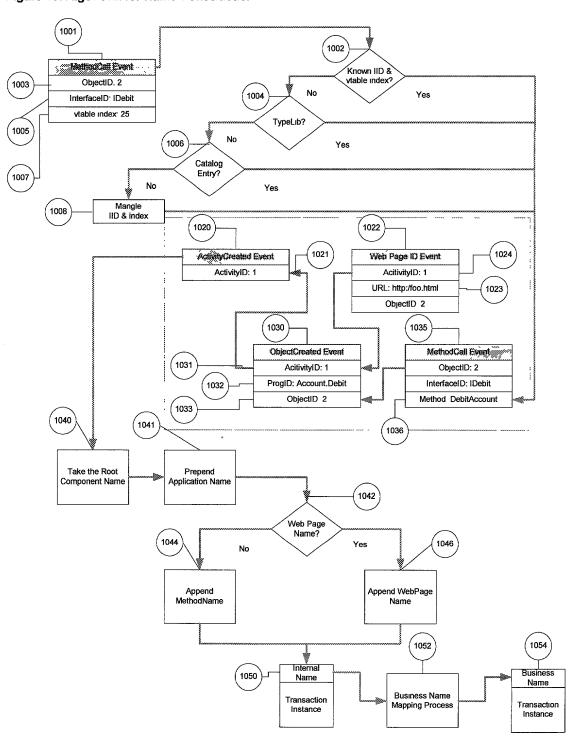
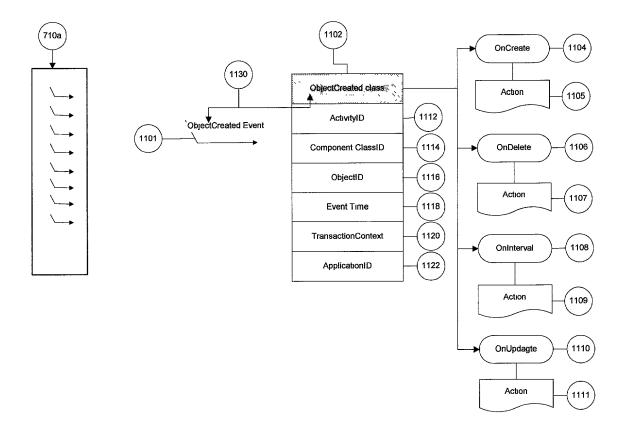


Figure 10: Algorithm for Name Construction

Figure 11:The Event Factory



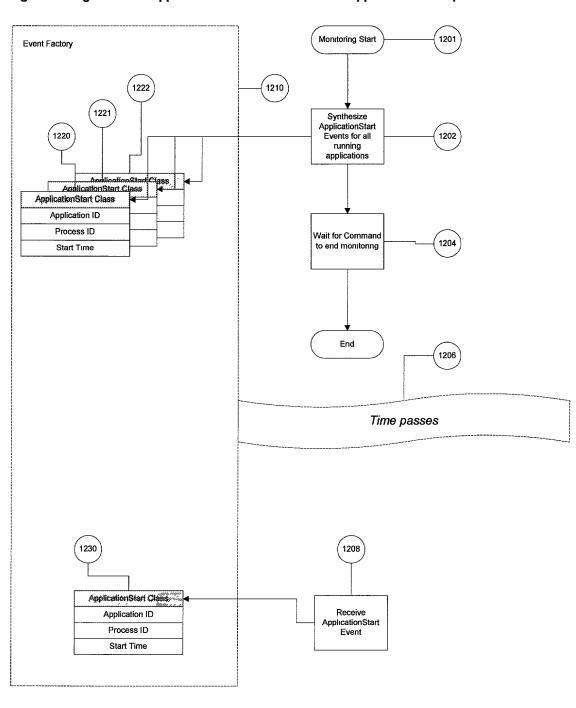


Figure 12: Algorithm for Application and Process Metrics - Application Startup

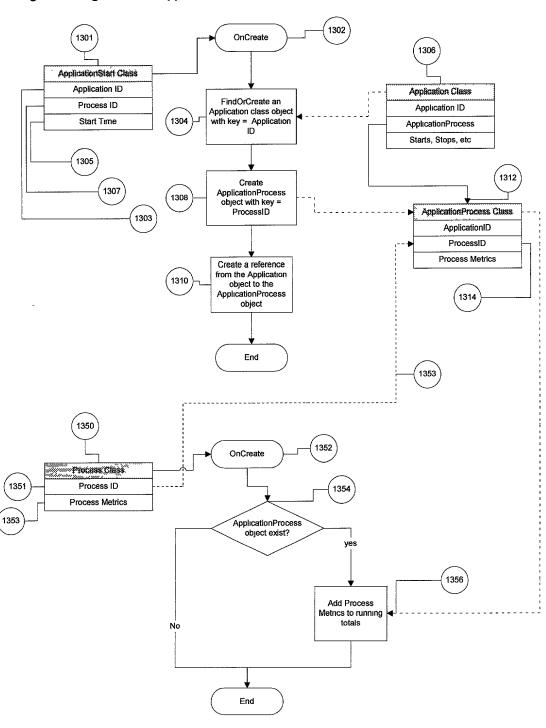


Figure 13: Algorithm for Application and Process Metrics

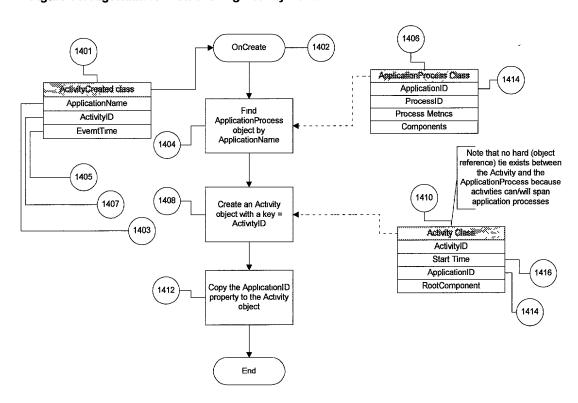
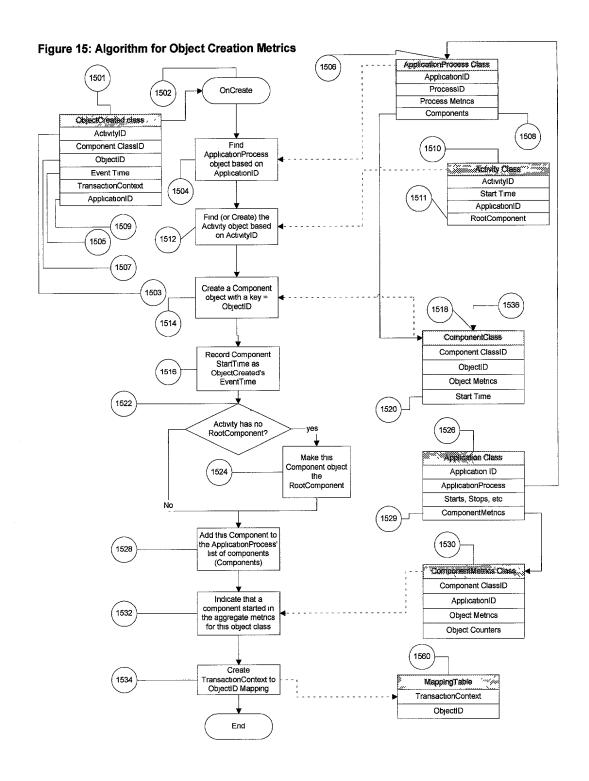
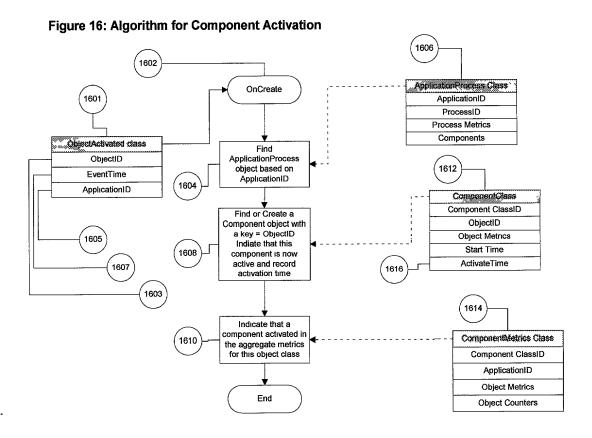
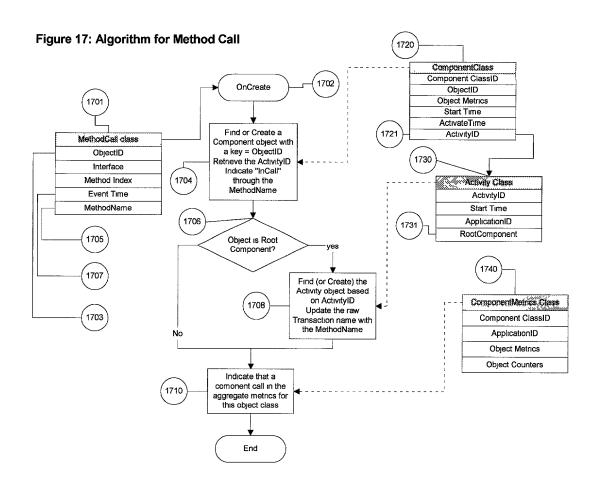
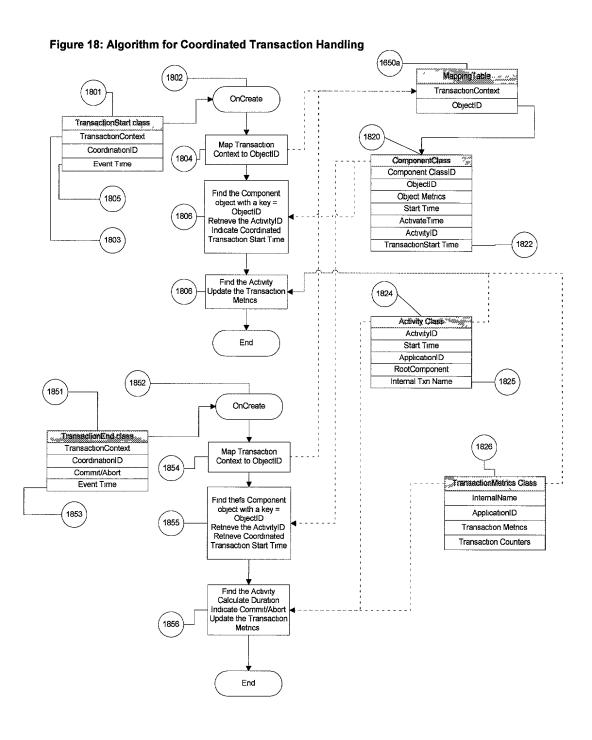


Figure 14: Algorithm for Establishing Activity Context









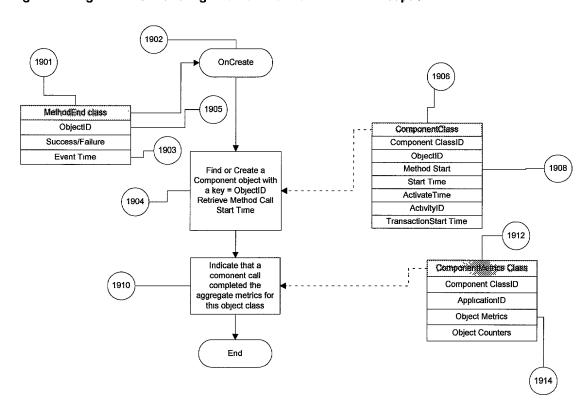


Figure 19: Algorithm for Handling Method Return and Method Exception

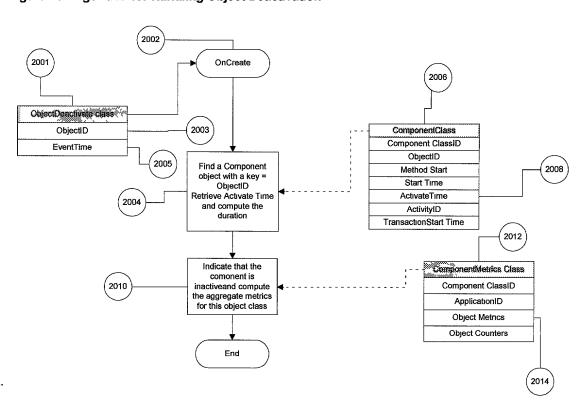


Figure 20: Algorithm for Handling Object Deactivation

2102 2101 OnCreate 2106 ObjectDestroyed class 2103 ObjectID ComponentClass Component ClassID EventTime ObjectID 2105 2108 Method Start Find or Create a Component object with Start Time a key = ObjectID ActivateTime 2104 Retrieve Start Time ActivityID Calculate the duration TransactionStart Time Destroy the 2110 Component object End 2120 2122 OnDestroy ComponentClass Component ClassID Log this Component? ObjectiD Method Start Start Time Write Component ActivateTime Metrics to Log file ActivityID TransactionStart Time 2128 No ComponentMetrics Class Update the aggregate mfsetrics for this object Component ClassID 2126 ApplicationID Object Metrics **Object Counters**

End

Figure 21: Algorithm for Handling Object Destruction

Application Class 2202 2201 2212 Application ID OnCreate ApplicationProcess Starts, Stops, etc ApplicationStop class FindOrCreate an Application ID Application class object with key = Application ID ApplicationProcess Class Process ID 2204 2206 ApplicationID **Event Time** ProcessID Success/Crash/Operator Compute duration from start, update all Process Metrics Components 2205 Application metrics, mark success/crash/ operator termination 2208 2207 2203 Delete ApplicationProcess objec t End 2252 2250 OnDestroy ApplicationProcess Class Iterate through the ComponentClass 2254 ApplicationID Components list, Component ClassiD ProcessID retrieve the Activity ID, destroy the Component Process Metrics ObjectID Components ActivityID Start Time (2258 Cleanup the Activity if the Component is the Activity's Root Component (2255 2260 Activity Class ActivityID 2256 Start Time Log Application Metrics ApplicationID to the Log File RootComponent 2262 End

Figure 22: Algorithm for Application Process Termination

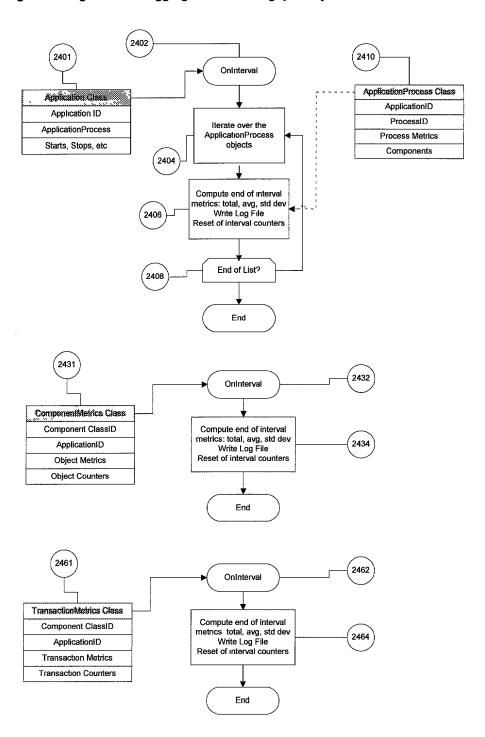


Figure 23: Algorithm for Aggregate and Throughput Object Metrics

FIG. 23